To appear in: Proceedings of the Fourth Connectionist Models Summer School
Lawrence Erlbaum Publisher, Hillsdale, NJ, Dec. 1993

Issues in Using Function Approximation for Reinforcement Learning

Sebastian Thrun
Institut für Informatik III
Universität Bonn
Römerstr. 164, D–53225 Bonn, Germany
thrun@cs.uni-bonn.de

Anton Schwartz
Dept. of Computer Science
Stanford University
Stanford, CA 94305
schwartz@cs.stanford.edu

Reinforcement learning techniques address the problem of learning to select actions in unknown, dynamic environments. It is widely acknowledged that to be of use in complex domains, reinforcement learning techniques must be combined with generalizing function approximation methods such as artificial neural networks. Little, however, is understood about the theoretical properties of such combinations, and many researchers have encountered failures in practice. In this paper we identify a prime source of such failures—namely, a systematic overestimation of utility values. Using Watkins' Q-Learning [18] as an example, we give a theoretical account of the phenomenon, deriving conditions under which one may expected it to cause learning to fail. Employing some of the most popular function approximators, we present experimental results which support the theoretical findings.

1 Introduction

Reinforcement learning methods [1, 16, 18] address the problem of learning, through experimentation, to choose actions so as to maximize one's productivity in unknown, dynamic environments. Unlike most learning algorithms that have been studied in the field of Machine Learning, reinforcement learning techniques allow for finding optimal action sequences in temporal decision tasks where the external evaluation is sparse, and neither the effects of actions, nor the temporal delay between actions and its effects on the learner's performance is known to the learner beforehand. The designated goal of learning is to find an *optimal policy*, which is a policy for action selection that maximizes future payoff (reward). In order to do so, most current reinforcement learning techniques estimate the *value* of actions, *i.e.*, the future payoff one can expect as a result of executing an action, using recursive estimation techniques.

In recent years, various theoretical results have been obtained that characterize the convergence properties of reinforcement learning algorithms. Strong stochastic convergence has been shown for a class of learning algorithms including Q-Learning, the most frequently used reinforcement learning technique [1, 6, 18, 19]. Others [7, 20] have characterized the convergence speed in terms of the size and other key characteristics of the domain at hand. While all these results apply exclusively to non-generalizing look-up table representations, it has often been argued, and seems to be generally accepted, that reinforcement learning techniques must be combined with powerful function approximators in order to scale to more complex tasks. This is because most domains studied in AI are too large to be searched exhaustively, hence demanding some kind of generalization. Generalization replaces costly training experience. Indeed, generalizing approximation techniques such as artificial neural networks and instance-based methods have been used in practice with some remarkable success in domains such as game playing [17, 2] and robotics [5, 8, 10]. For example, Tesauro [17] reports a backgammon computer program that reaches grand-master level of play, which has been constructed using a combination of reinforcement learning techniques and artificial neural networks.

Despite these encouraging empirical results, little is known about the general implications of using function approximation in reinforcement learning. Recently, Bradtke [3] has shown the convergence of a particular policy iteration algorithm when combined with a quadratic function approximator.[1] To our knowledge this is the only convergence proof for a reinforcement learning method using a generalizing function approximator to date. Singh and Yee [14] proposed a theoretical justification for using certain error-bounded classes of function approximators in the context of reinforcement learning.[2] Others, however, report failure in applying function approximators such as the Backpropagation algorithm [4, 8, 9]. In some cases learning failed since the function approximator at hand was not capable of representing reasonable value functions at all [13]. In other cases, however, failure was observed even though the function approximator at hand was able to represent a suitable value function, and thus a near-optimal policy could have been learned [3, 9]. In this paper we will focus our attention primarily on the latter cases. Obviously, there are inherent difficulties in combining function approximation and reinforcement learning techniques that do not exist for each component in isolation.

In the current paper we point out some significant pitfalls and limitations that arise from this combination. First, we present a theoretical analysis that elucidates effects that may occur when reinforcement learning is combined with a generalizing function approximator. Generally speaking, function approximators induce some noise (generalization error) on the output predictions. The key observation here is that such noise can lead to a *systematic overestimation effect* of values. As a consequence of this overestimation, as well as of the discounting typically used in reinforcement learning, the learner is *expected* to fail to learn an optimal policy in certain situations. We use this observation to derive practical bounds on (a) the necessary accuracy of the function approximator, and (b) the temporal discount factor used in learning. The theoretical analysis is followed by empirical results in a simple robot navigation domain using a variety of non-linear function approximators.

The goal of the papers is twofold. For one, we wish to contribute to the understanding of the effects that function approximation has in the context of reinforcement learning. In addition, we aim to elucidate practical pitfalls and to provide guidelines that might be helpful for actual implementations. This research has been driven by empirical difficulties that were encountered when combining various function approximators in a variety of domains, ranging from robot learning domains to learning in chess. The effects described in this paper match our personal experimental experience to a high degree, hence provide reasonable explanations for a multitude of previously unexplained findings.

2 The Overestimation Phenomenon

In this and the following section we present a theoretical analysis of systematic overestimation effects in reinforcement learning. Throughout this analysis we use Watkins' $\mathcal{Q}$-Learning as an example [18]. $\mathcal{Q}$-Learning is a technique for learning optimal policies in Markovian sequential decision tasks. It does this by incrementally learning a function $\mathcal{Q}(s,a)$ which it uses to evaluate the utility of performing action a in state s. More specifically, assume the agent observes during learning that action a executed at state s resulted in the state s' and some immediate reward r_s^a. This observation is employed to update $\mathcal{Q}$:

$$\mathcal{Q}(s,a) \quad \longleftarrow \quad r_s^a \; + \; \gamma \max_{\hat{a} \text{ action}} \mathcal{Q}(s',\hat{a}) \tag{1}$$

Here γ is a *discount factor*[3] ($0<\gamma<1$), used to give a preference for rewards reaped sooner in time. At any time, the $\mathcal{Q}$-values suggest a policy for choosing actions—namely the one which, in any state, chooses action a which maximizes $\mathcal{Q}(s,a)$. It has been shown that repeated application of this update equation eventually yields $\mathcal{Q}$-values that give rise to a policy which maximizes the expected cumulative discounted reward [18]. However, such results only apply when the $\mathcal{Q}$-values are stored precisely, *e.g.*, by a look-up table.

Assume, instead, that $\mathcal{Q}$ is represented by a *function approximator* that induces some noise on the estimates of $\mathcal{Q}$. More specifically, let us assume that the currently stored $\mathcal{Q}$-values, denoted by $\mathcal{Q}^{\text{approx}}$, represent some implicit target

[1] This proof addresses control problems solvable by *linear quadratic regulation*, a specific set of control tasks with linear dynamics and linear optimal policies.

[2] They derive worst-case bounds on the loss in performance in cases where, after learning, function approximation noise disturbs the optimal value function by some small amount.

[3] We ignore the degenerate case of $\gamma = 0$ to simplify the statement of the theorems.

(a) (b) (c)

Figure 1: Overestimation: (a) Q-values for six actions $a_1, \ldots, a_6$ are shown. If these Q-values are noise-free, the "max" will return the correct value Q^{target}. If there is noise, some overestimation of the maximum value is likely, as the expected return is $E[\max Q]$. (b) The expected overestimation is maximal if all Q-values are the same, and (c) minimal if the error interval of the best and the second best Q-value do not overlap (*cf.* the Corollary).

---

values Q^{target}, corrupted by a noise term $Y_{s'}^{\hat{a}}$ which is due to the function approximator.

$$Q^{\text{approx}}(s', \hat{a}) \quad = \quad Q^{\text{target}}(s', \hat{a}) \;+\; Y_{s'}^{\hat{a}}$$

Here the noise is modeled by the family of random variables $Y_{s'}^{\hat{a}}$ with zero mean. Clearly, this noise causes some error on the left-hand side of Eq. (1), denoted by the random variable Z_s:

$$\begin{aligned} Z_s &\stackrel{\text{def}}{=} r_s^a + \gamma \max_{\hat{a} \text{ action}} Q^{\text{approx}}(s', \hat{a}) - \left(r_s^a + \gamma \max_{\hat{a} \text{ action}} Q^{\text{target}}(s', \hat{a}) \right) \\ &= \gamma \left(\max_{\hat{a} \text{ action}} Q^{\text{approx}}(s', \hat{a}) - \max_{\hat{a} \text{ action}} Q^{\text{target}}(s', \hat{a}) \right) \end{aligned} \tag{2}$$

The key observation underlying our analysis is that *zero-mean noise* $Y_{s'}^{\hat{a}}$ may easily result in Z_s with *positive mean, i.e.*,

$$E[Y_{s'}^{\hat{a}}] = 0 \;\; \forall \hat{a} \quad \stackrel{\text{often}}{\Longrightarrow} \quad E[Z_s] > 0$$

To see this, consider the situation depicted in Fig. 1a, which illustrates a single update step according to Eq. (1). In this example, as well as in our theoretical analysis, the noise variables $Y_{s'}^{\hat{a}}$ are modeled by independent, uniformly distributed random variables in the interval $[-\varepsilon, \varepsilon]$ (for some $\varepsilon > 0$), as indicated by the error bars. Now consider the calculation of the maximum in Eq. (1), assuming that there is more than one action to choose from. Due to the function approximation noise, some of the Q-values might be too small, while others might be too large. The max operator, however, always picks the largest value, making it particularly sensitive to overestimations. If several Q-values are alike and their error intervals overlap, one is likely to overestimate the correct Q-value for some action, which will make the max operator overestimate as well. In short, max causes overestimation because it does not preserve the zero-mean property of the errors of its operands.

We now address the question of how large the expected overestimation $E[Z_s]$ may be.

Lemma. *Let n denote the number of actions applicable at state s'. If all n actions share the same target Q-value,* i.e., $\exists q : \forall \hat{a} : q = Q^{\text{target}}(s', \hat{a})$*, then the average overestimation $E[Z_s]$ is γc with $c \stackrel{\text{def}}{=} \varepsilon \frac{n-1}{n+1}$.*

The proof of this Lemma, as well as those of the theorems below, can be found in the Appendix. This Lemma demonstrates that each time the update rule (1) is applied, the expected overestimation can be as large as γc, depending on the Q-values at the state s'. Hence:

Corollary. $0 \leq E[Z_s] \leq \gamma c$ *with* $c = \varepsilon \frac{n-1}{n+1}$.

The particular expected overestimation $E[Z_s]$ depends on the environment at hand. In many fine-grained navigation domains a variety of movement actions may result in transition to locations of approximately equal merit—in such cases the actions will have roughly equal Q-values, and we may expect $E[Z_s]$ to be close to γc (*cf.* Fig.1b). In game playing domains we expect $E[Z_s]$ to be closer to 0 (*cf.* Fig.1c), since often the number of good moves is small.

3 Bounds for Expected Failure of Q-Learning

Thus far, we have looked at the error induced by *single* applications of the update rule. In what follows, we will explore the effects of systematic overestimation on Q-Learning, in which the update rule is applied iteratively.

For the sake of simplicity, let us assume (a) that there is a set of goal states, (b) positive reward r_{goal} is only received upon entering a goal state, (c) $r_{\text{goal}} = 1$, and (d) the state-transition function is deterministic.[4] Consider an optimal state-action sequence $\langle s_i, a_i \rangle$ ($i \in \{0, \ldots, L\}$) from some initial state, denoted by s_0, to the nearest goal state. One necessary condition for the the success of Q-Learning (except for some degenerate cases) is that the sequence of Q-values $Q(s_i, a_i)$ is monotonically increasing in i:

$$Q(s_i, a_i) \quad \leq \quad Q(s_{i+1}, a_{i+1}) \qquad \text{for all } i \in \{0, \ldots, L-1\} \tag{3}$$

It is easy to see that a violation of the monotonicity condition may result in a wrong ordering of utilities, which generally precludes optimal action choice. We will now consider two more concrete "worst-case" scenarios.

Case 1. Assume that the learner *always* overestimates Q-values by γc. In order to ensure the monotonicity, the discount factor γ must compensate for this. This observation leads to the first theorem:

Theorem 1. *If there is maximal, repeated overestimation of magnitude γc along an optimal path, Q-Learning is expected to fail to learn an optimal policy if $\gamma > \frac{1}{1+c}$.*

Case 2. Now assume that Q-Learning managed to learn the last $L-1$ Q-vales of this optimal path *correctly*—Depending on the task at hand, this might nor might not be what one would expect (*cf.* the Corollary), but this is the designated goal of learning[5]. Then these Q-values are given by iteratively discounting the final reward with the distance to the goal state, *i.e.*, $Q(s_{L-i}, a_{L-i}) = \gamma^i$ for $i \in \{1, \ldots, L-1\}$. Consequently, the *correct* Q-value $Q^{\text{correct}}(s_0, a_0)$ is γ^L. What happens if noise in function approximation results in an expected overestimation γc when updating this value? In order to ensure the monotonicity of Q, we have to make sure that the difference between $Q(s_1, a_1)$ and $Q(s_0, a_0)$ is at least γc, *i.e.*,

$$\gamma^{L-1} - \gamma^L \quad \geq \quad \gamma\ c. \tag{4}$$

This leads to the following theorems, which represent average-case analyses for learning paths of length L in such "worst-case" environments.

Theorem 2. *Under the conditions stated above, Q-Learning is expected to fail if $\varepsilon > \frac{n+1}{n-1} \cdot \frac{(L-2)^{L-2}}{(L-1)^{L-1}}$ $\left(\approx \frac{1}{L}\right)$.*

Table 1 shows upper bounds on ε for some choices of L and n. It can be seen that for reasonably large L and n the function approximator must be extremely accurate, if Q-Learning is to successfully identify optimal policies.

The bound in Theorem 2 does not depend on γ. Given that ε is small enough to fulfill the condition in Theorem 2, condition (4) also establishes bounds on the choice of the discount factor γ.

Theorem 3. *Under the conditions stated above, Q-Learning is expected to fail if $\gamma^{L-1} - \gamma^L < \gamma\ c$.*

Table 2 shows some bounds on γ as a function of ε and L. Note that Theorem 1 is a special case of Theorem 3 for $L = 1$. For larger values of L the bounds are tighter. For example, if $L \geq 2$, Theorem 3 implies that γ must be smaller than $1 - c = 1 - \varepsilon\frac{n-1}{n+1}$. If $L \geq 3$, γ must be between $.5-\sqrt{.25 - c}$ and $.5+\sqrt{.25 - c}$.

[4] Assumption (c) is made without loss of generality, if we assume that the error ϵ scales linearly with the range of values to be represented. Assumption (d) is inessential to the analysis but is adopted to simplify the presentation here. (a) and (b) are restrictive, but describe a large number of the tasks addressed in the literature.

[5] Surprisingly, this turns out to be a worst case scenario for this analysis.

Table 1: Upper bound on the error ε of the function approximator, according to Theorem 2. These bounds are significant. For example, if episodes of length $L = 60$ with $n = 5$ actions shall be learned, ε must be smaller than .00943 (bold number).

	L=10	L=20	L=30	L=40	L=50	L=60	L=70	L=80	L=90	L=100	L=1000
$n = 2$	.12991	.05966	.03872	.02866	.02275	.01886	.01611	.01405	.01247	.01120	.00110
$n = 3$	.08660	.03977	.02581	.01911	.01517	.01257	.01074	.00937	.00831	.00746	.00073
$n = 4$	.07217	.03314	.02151	.01592	.01264	.01048	.00895	.00781	.00692	.00622	.00061
$n = 5$	.06495	.02983	.01936	.01433	.01137	**.00943**	.00805	.00702	.00623	.00560	.00055
$n = 6$	.06062	.02784	.01807	.01337	.01061	.00880	.00751	.00656	.00581	.00522	.00051
$n = 8$	.05567	.02557	.01659	.01228	.00975	.00808	.00690	.00602	.00534	.00480	.00047
$n = 10$	.05292	.02430	.01577	.01167	.00927	.00768	.00656	.00572	.00508	.00456	.00045
$n = 20$	.04786	.02198	.01426	.01056	.00838	.00695	.00593	.00517	.00459	.00412	.00040
$n = \infty$	.04330	.01988	.01290	.00955	.00758	.00628	.00537	.00468	.00415	.00373	.00036

Table 2: Upper and lower bounds for the choice of γ due to Theorem 3 for the fixed number of actions $n = 5$. Open slots indicate that no γ is expected to work at all (*cf.* Theorem 2). For example, function approximators with $\varepsilon = .05$ are unlikely to learn optimal action sequences longer than $L = 11$. $\gamma = .7$ diminishes the maximum length L to 8.

	$\varepsilon = .3$	$\varepsilon = .2$	$\varepsilon = .15$	$\varepsilon = .125$	$\varepsilon = .1$	$\varepsilon = .075$	$\varepsilon = .05$	$\varepsilon = .025$
$L = 1$	.0000/.8333	.0000/.8823	.0000/.9090	.0000/.9230	.0000/.9375	.0000/.9523	.0000/.9677	.0000/.9836
$L = 2$	.0000/.8000	.0000/.8666	.0000/.9000	.0000/.9166	.0000/.9333	.0000/.9500	.0000/.9666	.0000/.9833
$L = 3$	.2763/.7236	.1584/.8415	.1127/.8872	.0917/.9082	.0718/.9281	.0527/.9472	.0345/.9654	.0169/.9830
$L = 4$	—	.5361/.7819	.4126/.8669	.3611/.8962	.3110/.9214	.2599/.9438	.2047/.9641	.1391/.9827
$L = 5$	—	—	.6753/.8158	.5861/.8760	.5166/.9121	.4495/.9397	.3767/.9626	.2857/.9824
$L = 6$	—	—	—	—	.6708/.8970	.5914/.9344	.5109/.9609	.4099/.9820
$L = 7$	—	—	—	—	.8182/.8477	.6978/.9269	.6121/.9588	.5081/.9817
$L = 8$	—	—	—	—	—	.7829/.9145	**.6893/.9564**	.5852/.9813
$L = 9$	—	—	—	—	—	—	.7497/.9534	.6463/.9809
$L = 10$	—	—	—	—	—	—	.7986/.9495	.6954/.9804
$L = 11$	—	—	—	—	—	—	.8401/.9440	.7355/.9800
$L = 12$	—	—	—	—	—	—	**.8791/.9340**	.7687/.9794
$L = 13$	—	—	—	—	—	—	—	.7965/.9789
$L = 14$	—	—	—	—	—	—	—	.8202/.9783
$L = 15$	—	—	—	—	—	—	—	.8405/.9776
⋮								⋮
$L = 23$	—	—	—	—	—	—	—	.9436/.9640
$L \geq 24$	—	—	—	—	—	—	—	—

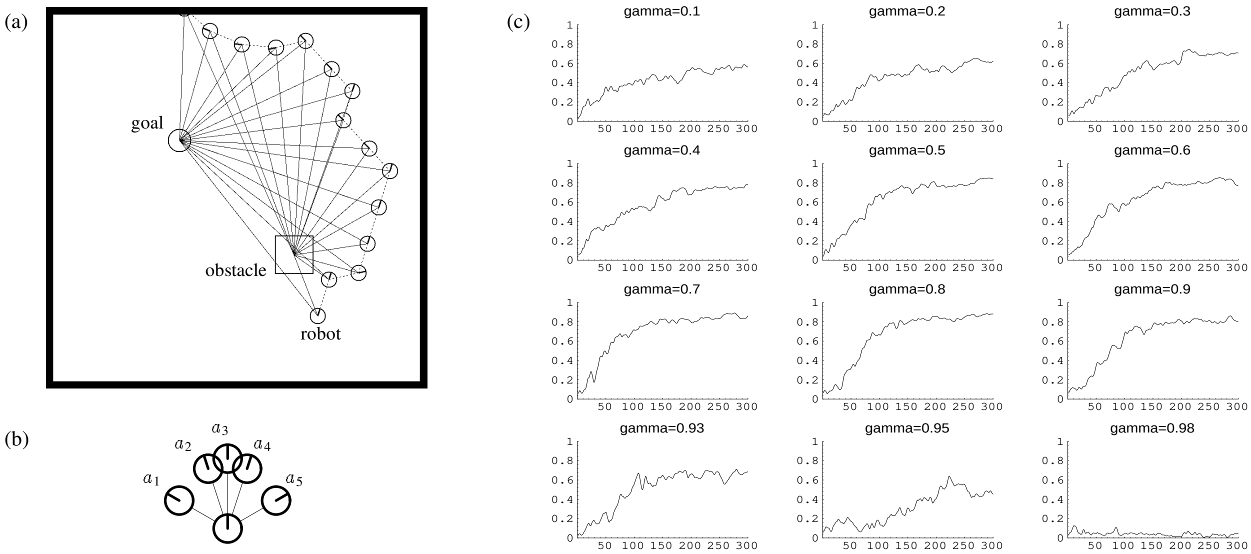

Figure 2: (a) The simulated robot domain. (b) Action space. (c) Learning curves as a function of γ. Each diagram shows the performance (probability of reaching the goal state) as a function of the number of training episodes. The performance is evaluated on an independent testing set of twenty initial robot positions. All curves are averaged over eight runs. Note that learning fails completely if $\gamma \geq .98$.

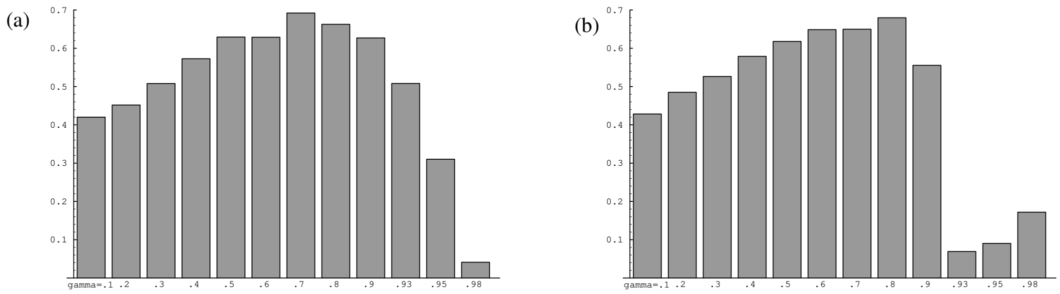

Figure 3: Performance as a function of γ, averaged over the first 300 episodes of learning (eight runs each). (a) Instance-based learning and (b) plain Backpropagation (sigmoidal transfer functions).

4 Empirical Results

The theoretical results were empirically validated using the simulated robot environment depicted in Fig. 2a (*cf.* [9]). The task is to learn a policy that carries the robot agent from arbitrary starting positions to the goal. The robot can sense the state it is in using sensors which measure the distance and orientation of both the goal and the obstacle. It has five actions to chose from, as depicted in Fig. 2b. Positive reward $+1$ is received upon entering the goal region, while collisions are penalized by a reward of -1.

Six series of simulations were performed using (a) instance-based learning with local polynomial approximation, as described in [9, 11], and Backpropagation learning [12] with (b) standard sigmoidal, (c) sinusoidal and (d) radial basis transfer functions. In two further experiments we used radial basis functions for the hidden units only, together with (e) sigmoidal and (f) linear output units. In all experiments the Q-functions for the different actions were represented by separate function approximators. In order to exploit the training information maximally, we used an off-line replay technique similar to that reported in [8]. Parameters were optimized individually for the different approximation techniques.

In a systematic comparison, (a) was empirically found to work best for the task at hand, followed closely by (b) and (e). (f) managed to archive a mediocre level of performance in the given time for a small range of discount factors, and (c) and (d) failed completely to learn a policy that resulted in a better-than-random performance. Figure 2c shows learning curves for the instance-based function approximator (a). Summaries for the techniques (a) and (b) are depicted in Figure 3. These results illustrate the existence of practical bounds on γ as predicted by Theorems 1 and 3. In the less successful cases, visual inspection of the Q-values during learning showed that overestimation prevented these techniques from learning a reasonable policy. For example, in (f) we frequently observed that the learned values exceeded the corresponding target values by a difference of 10 or more, even though the maximum final reward 1 establishes an upper bound on the values. Theorem 2 gives a reasonable explanation for these findings.

5 Discussion

In this paper we have pointed out pitfalls of combining reinforcement learning with function approximators. In particular, we described a *systematic overestimation effect* of values which is due to function approximation when used in recursive value estimation schemes. We analyzed this effect to derive bounds on (a) the necessary accuracy of the function approximator ε, and (b) the discount factor γ. An empirical comparison of different function approximators illustrated the phenomena presented in the theoretical analysis.

Both the analysis and the experimental results give rise to the following potential strategies for diminishing the catastrophic effects of overestimation.

- The results can be interpreted as an argument in favor of approximators with *unbounded memory*. The error of instance-based approximation techniques, such as the one we used in the experiments, may approach zero as the number of training examples goes to infinity. Therefore, we would expect that the overestimation gradually vanishes over time. This seems implausible to assume for approximators with bounded memory, such as the

Backpropagation algorithm. Approximators with bounded memory are often likely to have some minimum residual error, unless specific information about the environment is available that ensures that the error can go to zero.

- Reinforcement learning techniques based on $TD(\lambda)$ [15] update values according to a multitude of values and observed rewards. Using a $TD(\lambda)$ scheme with $\lambda > 0$ should reduce the effects of overestimation, since incorporating actual sample rewards rather than (over)estimated values in the update equation will give rise to less overestimation.

- The analysis suggests a move from discounted to undiscounted frameworks, in which additive rather than multiplicative costs are applied at every step. Recall that temporal discounting, which imposes a cost term that is applied multiplicatively, makes differences in neighboring values exponentially decrease, as the distance to the goal state increases. This observation played a crucial role in Sect. 3, where we derived bounds on ε and γ. With additive costs, the differences in neighboring values do not go to zero as distance to the goal increases, but remain constant, preserving the results of Sect. 2 but not of Sect. 3. It remains to be seen in practice, though, to what extent the expected failure discussed in the paper can be reduced by using costs instead of discounting.

- The effects of overestimation can also be diminished by introducing pseudo-costs to offset overestimation. Given that one knows enough about the domain and the function approximator at hand to estimate the overestimation which occurs, one could impose that estimate as an additional cost on actions to compensate for the effect. Pseudo-costs may be used in either discounted or undiscounted scenarios.

- Another strategy to diminish the effects of overestimation is to introduce function approximators that are biased toward making low predictions. For example, in instance-based learning techniques one could bias the learner to predict low values whenever the data points are sparse. Similar effects can be achieved for Backpropagation-type algorithms, if one trains on additional, synthetic training points that have a low target value. Both techniques will encourage approximators to generate low predictions on regions with sparsely distributed training points. In general, such modifications decrease the likelihood of overestimation by increasing the likelihood of underestimation, which reduces the amount of systematic overestimation.

When drawing conclusions from the theoretical analysis, one must be aware of the limiting assumptions made therein. In particular, we assumed uniform distributed, independent generalization error, and addressed a restricted class of deterministic domains. We based our analysis on the Q-Learning algorithm with zero costs and temporal discounting. However, many of these assumptions were violated in the experimental setting. Although the theoretical analysis addresses exclusively the learning of optimal policies, the results seem to carry over to learning good approximations as well. We suspect that the effects described in this paper carry over to other dynamic programming-related reinforcement learning techniques, to other models of function approximators, to mutually dependent errors[6], and to stochastic domains. These effects, however, address only one facet of using function approximation in the context of reinforcement learning; clearly, the full consequences of this combination are far more complex, and are the subject of ongoing research.

Acknowledgment

We thank Peter Dayan for his thoughtful comments on an earlier version of this paper. We also thank Matthew McDonald for making us aware of two typos in the proof section of this paper.

[6] It should be noted that errors in function approximation are generally not independent. Dependencies are likely to affect the amount of overestimation, but prove difficult to quantify and analyze.

References

[1] A. G. Barto, S. J. Bradtke, and S. P. Singh, "Learning to act using real-time dynamic programming," *Artificial Intelligence*, 1993. (to appear).

[2] J. A. Boyan, "Modular neural networks for learning context-dependent game strategies," Master's thesis, University of Cambridge, UK, August 1992.

[3] S. J. Bradtke, "Reinforcement learning applied to linear quadratic regulation," in *Advances in Neural Information Processing Systems 5* S. J. Hanson, J. Cowan, and C. L. Giles, eds., San Mateo, CA, pp. 295–302, Morgan Kaufmann, 1993.

[4] D. Chapman and L. P. Kaelbling, "Input generalization in delayed reinforcement learning: an algorithm and performance comparisons," in *Proceedings of IJCAI-91*, Darling Habour, Sydney, Australia, IJCAI, Inc., 1991.

[5] V. Gullapalli, *Reinforcement Learning and its Application to Control*. PhD thesis, Department of Computer and Information Science, University of Massachusetts, 1992.

[6] T. Jaakkola, M. I. Jordan, and S. P. Singh, "On the convergence of stochastic iterative dynamic programming algorithms," Tech. Rep. 9307, Department of Brain and Cognitive Sciences, Massachusetts Institut of Technology, July 1993.

[7] S. Koenig and R. G. Simmons, "Complexity analysis of real-time reinforcement learning applied to finding shortest paths in deterministic domains," Tech. Rep. CMU-CS-93-106, Carnegie Mellon University, December 1992.

[8] L.-J. Lin, *Self-supervised Learning by Reinforcement and Artificial Neural Networks*. PhD thesis, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA, 1992.

[9] T. M. Mitchell and S. B. Thrun, "Explanation-based neural network learning for robot control," in *Advances in Neural Information Processing Systems 5* S. J. Hanson, J. Cowan, and C. L. Giles, eds., San Mateo, CA, pp. 287–294, Morgan Kaufmann, 1993.

[10] A. W. Moore, *Efficient Memory-based Learning for Robot Control*. PhD thesis, Trinity Hall, University of Cambridge, England, 1990.

[11] A. W. Moore and C. G. Atkeson, "Memory-based function approximators for learning control," MIT AI-Lab Memo, July 1992.

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing. Vol. I + II* D. E. Rumelhart and J. L. McClelland, eds., MIT Press, 1986.

[13] P. Sabes, "Q-learning with a basis function representation for the Q-values," same volume.

[14] S. P. Singh and R. C. Yee, "An upper bound on the loss from approximate optimal-value functions," (submitted for publication), 1993.

[15] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, 1988.

[16] R. S. Sutton, *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, Department of Computer and Information Science, University of Massachusetts, 1984.

[17] G. J. Tesauro, "Practical issues in temporal difference learning," *Machine Learning Journal*, vol. 8, 1992.

[18] C. J. C. H. Watkins, *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, England, 1989.

[19] C. J. Watkins and P. Dayan, "Q-learning," *Machine Learning Journal*, vol. 8, pp. 279–292, 1992.

[20] S. D. Whitehead, "A study of cooperative mechanisms for faster reinforcement learning," Tech. Rep. 365, University of Rochester, Computer Science Department, Rochester, NY, March 1991.

Appendix: Proofs

Lemma. Let $f(x)$ denote the density of the noise variables $Y_{s'}^{\hat{a}}$, *i.e.*, $f(x) \stackrel{def}{=} Prob[Y_{s'}^{\hat{a}} = x] = \frac{1}{2\varepsilon}$. Then

$$
\begin{aligned}
E[Z_s] &\stackrel{(2)}{=} E\left[\gamma\left(\max_{\hat{a}\text{ action}} \mathcal{Q}^{\text{approx}}(s',\hat{a}) - \max_{\hat{a}\text{ action}} \mathcal{Q}^{\text{target}}(s',\hat{a})\right)\right] \;=\; \gamma\, E\left[\max_{\hat{a}\text{ action}} Y_{s'}^{\hat{a}}\right] \\
&= \gamma \int_{-\infty}^{\infty} x\; n \underbrace{f(x)}_{Prob[Y=x]} \Bigg(\underbrace{\int_{-\infty}^{x} f(z)\; dz}_{Prob[Y\leq x]}\Bigg)^{n-1} dx \;=\; \gamma\, n \int_{-\varepsilon}^{\varepsilon} x\, \frac{1}{2\varepsilon}\left(\frac{1}{2}+\frac{x}{2\varepsilon}\right)^{n-1} dx \\
&= \gamma\, n \int_0^1 (2\varepsilon y - \varepsilon) y^{n-1}\; dy \qquad\qquad \text{(by substituting } y = \frac{1}{2}+\frac{x}{2\varepsilon}) \\
&= \gamma\, n\, \varepsilon \int_0^1 2y^n - y^{n-1}\; dy \;=\; \gamma\, n\, \varepsilon\left(\frac{2}{n+1}-\frac{1}{n}\right) \;=\; \gamma\, \underbrace{\varepsilon\frac{n-1}{n+1}}_{=:\,c} \;=\; \gamma\, c
\end{aligned}
$$

□

Theorem 1. Let $q_i \stackrel{def}{=} E[\mathcal{Q}(s_i,a_i)]$ (for $i \in \{0,\ldots,L\}$) under the conditions stated in Theorem 1. Then $q_L = 1$ and $q_i = \gamma(q_{i+1}+c)$. It is easy to show by induction that

$$
q_i \;=\; \gamma^{L-i} + \sum_{k=1}^{L-i} \gamma^k c
$$

Using this closed-form expression, condition (3) can be re-written as

$$
\begin{aligned}
0 &\stackrel{(3)}{\geq} q_i - q_{i+1} \;=\; \gamma^{L-i} + \sum_{k=1}^{L-i}\gamma^k c \;-\; \gamma^{L-i-1} - \sum_{k=1}^{L-i-1}\gamma^k c \;=\; \gamma^{L-i-1}(\gamma-1)+\gamma^{L-i}c \\
&\stackrel{\gamma>0}{\iff} 0 \;\geq\; \gamma - 1 + \gamma c \;=\; \gamma(1+c)-1 \;\iff\; \gamma \;\leq\; \frac{1}{1+c}
\end{aligned}
$$

□

Theorem 2. Condition (4) can be re-written as

$$
\gamma^{L-1}-\gamma^L \stackrel{(4)}{\geq} \gamma\, c \;\iff\; \gamma^{L-1}-\gamma^L-\gamma\, c \;\geq\; 0 \;\stackrel{\gamma>0}{\iff}\; \underbrace{\gamma^{L-2}-\gamma^{L-1}-c}_{(*)} \;\geq\; 0 \tag{5}
$$

The left-hand expression $(*)$ takes its maximum at $\gamma^* = \frac{L-2}{L-1}$, since the first derivative of $(*)$

$$
\frac{\partial}{\partial\gamma}\gamma^{L-2}-\gamma^{L-1}-c \;=\; (L-2)\gamma^{L-3}-(L-1)\gamma^{L-2} \;=\; -(L-1)\gamma^{L-3}\left(\gamma-\frac{L-2}{L-1}\right) \tag{6}
$$

is 0 and $(*)$ is concave at $\gamma = \gamma^*$. Hence

$$
\begin{aligned}
0 &\stackrel{(5)}{\leq} \gamma^{L-2}-\gamma^{L-1}-\; c \;<\; \gamma^{*\,L-2}-\gamma^{*\,L-1}-c \;\stackrel{(6)}{=}\; \left(\frac{L-2}{L-1}\right)^{L-2} - \left(\frac{L-2}{L-1}\right)^{L-1} - c \\
&= \frac{(L-2)^{L-2}[(L-2)+1]-(L-2)^{L-1}}{(L-1)^{L-1}} - c \;=\; \frac{(L-2)^{L-2}}{(L-1)^{L-1}} - c \\
\iff c &\leq \frac{(L-2)^{L-2}}{(L-1)^{L-1}} \;\stackrel{c=\varepsilon\frac{n-1}{n+1}}{\iff}\; \varepsilon \;\leq\; \frac{n+1}{n-1}\cdot\frac{(L-2)^{L-2}}{(L-1)^{L-1}} \;\approx\; \frac{n+1}{(n-1)\,L} \;\approx\; \frac{1}{L}
\end{aligned}
$$

□